HZ BOOKS
华章教育

计算机基础课程系列教材

VISUAL C++ PROGRAMMING THIRD EDITION

Visual C++教程

第3版

郑阿奇 主编
丁有和 编著

机械工业出版社
China Machine Press

计算机基础课程系列教材

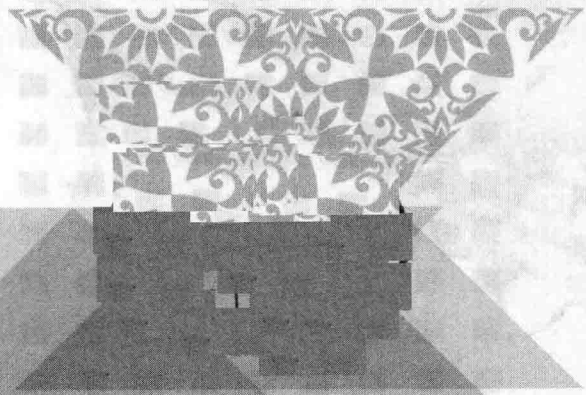
VISUAL C++ PROGRAMMING THIRD EDITION

Visual C++教程

第3版

郑阿奇 主编

丁有和 编著



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Visual C++教程 / 郑阿奇主编. —3版. —北京：机械工业出版社，2015.3
（计算机基础课程系列教材）

ISBN 978-7-111-49143-9

Ⅰ. V… Ⅱ. 郑… Ⅲ. C语言－程序设计－高等学校－教材 Ⅳ. TP312

中国版本图书馆CIP数据核字（2015）第012222号

本书在简单介绍C/C++的基础上，重点介绍Visual C++ 6.0程序设计，分为教程、实验与实习三个部分，主要包括“C/C++语言概述”、“C++面向对象程序设计基础”、“C++面向对象程序设计进阶”、“MFC框架、消息和对话框”、“常用控件”、“框架窗口界面设计”、“数据、文档和视图”、“图形、文本和数据库”等内容。在实践环节，配备15个实验进行知识块训练，最后进行综合实习，使知识块形成知识体，同时锻炼读者解决问题的能力。

本书基本每章包含常见问题解答、习题和单元综合测试，在书后配套单元综合测试参考答案。学期结束，学生可通过提供的两份模拟试卷和参考答案自测C++和Visual C++部分内容，教师可据此生成正式考试试卷。

通过阅读本书并且进行相关实验和综合实习，能在较短的时间内基本掌握Visual C++及其应用技术。本书可作为高等院校本科、高职高专学生的教材，也可作为广大Visual C++ 6.0用户的学习和参考用书。

出版发行：机械工业出版社（北京市西城区百万庄大街22号 邮政编码：100037）
责任编辑：余 洁 责任校对：董纪丽
印 刷：北京瑞德印刷有限公司 版 次：2015年3月第3版第1次印刷
开 本：185mm×260mm 1/16 印 张：21.5
书 号：ISBN 978-7-111-49143-9 定 价：40.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991 88361066 投稿热线：(010) 88379604
购书热线：(010) 68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前　言

在学习完C语言后再学习C++和Visual C++已经成为许多高等学校的一种模式。我们在汲取以前编写C++和Visual C++教材成功经验的基础上，结合C++和Visual C++的教学实践，以读者初步熟悉C++语言为前提，编写了《Visual C++教程》第1版和第2版，该书受到广大师生的好评，至今已经重印多次。

第3版在第2版的基础上进行修订，在简单介绍C/C++的基础上，重点介绍Visual C++ 6.0程序设计，分为教程、实验与实习3个部分。

对于C++基础内容部分，根据近年教学实践进行了完善。同时，加强了“实验1”的内容。

对于MFC部分，则做了如下调整。

第4章对Windows和MFC编程的框架内容进行细化，强化从本质上理解MFC的机制，同时对“对话框”内容进行优化。

第5章的常用控件去掉了“标签控件”内容，同时对内容做适当调整和优化。

第6章将整个框架的界面内容（窗口、图标、光标、菜单、工具栏和状态栏）的次序做了调整：在图标和光标后介绍菜单、工具栏和状态栏，同时将分散的菜单内容合并在一起，保持其相对独立。

第7章强调数据流的主线，去掉了“使用多个文档类型”，添加“类序列化”等内容。

第8章将图形、文本和数据库放在一起讲，去掉了“ADO数据库编程”，既强调了MFC ODBC数据库的应用，又避免了应用技术的复杂性。

另外，基本每章都配有常用问题解答、习题、单元综合测试等内容，以更好地为教学服务。

本书不仅适合教学使用，也非常适合利用Visual C++ 6.0编程和开发应用程序的用户学习和参考。通过阅读本书，结合上机操作指导进行练习，读者就能在较短的时间内基本掌握Visual C++及其应用技术。

本书主要由丁有和（南京师范大学）编写，郑阿奇（南京师范大学）对全书进行统编定稿。参加本书编写的还有顾韵华、梁敬东、朱毅华、时跃华、赵青松、彭作民、崔海源、徐卫军、刘毅、王燕平、汤玫、郑进、周怡君、刘博宇、吴明祥等。

本书为教师免费提供教学课件、实例源文件、综合应用实习源文件、模拟试卷、参考答案等教辅，需要者可从华章网站（www.hzbook.com）下载。

由于编者水平有限，不当之处在所难免，恳请读者批评指正。

编　者

2015年1月

目　录

第二部分 实 验

第三部分 实 习

第一部分　教　程

第 1 章　C/C++语言概述

最早的 C 语言是在 1970 年由 AT&T 贝尔实验室推出的。尽管 C 语言具有简洁高效、运算符丰富、兼有低级语言的特征和良好的可读性等许多优点，但为了能满足运用面向对象方法开发软件的需求，1983 年贝尔实验室对 C 语言进行了扩充和完善，开发出支持面向对象程序设计的 C++语言。为了使 C++具有良好的可移植性，C++的第一个国际标准 ISO/IEC 14882:1998 在 1998 年获得批准，这个标准常称为 C++98、标准 C++或 ANSI/ISO C++。C++第二版的标准（ISO/IEC 14882:2003）是在 2003 年发布的。随后，2011 年发布了 ISO/IEC 14882:2011，它是目前最新的标准，简称 C++11。但本书仍以 C++98 为基础。

本章主要简述 C/C++的基础内容，同时，凡 C++对 C 语言扩展的内容将予以注明。需要说明的是，在学习本章内容之前最好先做实验 1。

1.1　从C到C++的程序结构

和 C 语言一样，一个 C++程序也是由预处理命令、语句、函数、变量（对象）、输入与输出以及注释等几个基本部分组成的。下面先来看一个简单的 C++程序（注意与 C 语言的区别）。

【例 Ex_Simple】一个简单的 C++程序

```
// C++程序的基本结构
#include <iostream.h>
void main()
{
    double r, area;                          // 声明变量
    cout<<"输入圆的半径: ";                  // 显示提示信息
    cin>>r;                                  // 从键盘上输入变量r的值
    area = 3.14159 * r * r;                  // 计算面积
    cout<<"圆的面积为: "<<area<<"\n";        // 输出面积
}
```

该程序经编译、连接、运行后，屏幕上显示：

```
输入圆的半径:
```

此时等待用户输入，当输入“10”并按 Enter 键后，屏幕显示：

```
圆的面积为: 314.159
```

这就是程序运行的过程。

和 C 语言一样，代码中的 main 表示主函数，每一个 C++程序都必须包含一个且只能包含一个 main 函数。main 函数体是用一对花括号“{”和“}”括起来的，函数体中包括若干条语句，每一条语句都以分号“;”作为结束的标志。

在本例中，main 函数体的第 1 条语句用来声明 r 和 area 两个变量；第 2 条语句是一条输出语句，它将双引号中的内容输出到屏幕上，cout 表示标准输出流对象，用于屏幕输出，“<<”是

插入符，它将后面的内容插入到 cout 中，即输出到屏幕上；第 3 条语句是一条输入语句，cin 表示标准输入流对象，用于键盘输入，“>>”是提取符，用来将用户键入的内容保存到后面的变量 r 中；最后一条语句是采用多个“<<”将字符串和变量 area 的内容输出到屏幕中，后面的“\n”是换行符，即在输出内容后回车换行。

程序的第 2 行“#include <iostream.h>”是 C++的编译指令，称为预处理命令。iostream.h 文件是一个标准输入/输出流的头文件，由于程序中用到了输入/输出流对象 cin 和 cout，故需要包含该头文件。

从代码中可以看出，C++用标准输入输出的头文件 iostream.h 替代了 C 语言的 stdio.h，用 cin、cout 和操作运算符>>、<<等实现并扩展了 C 语言的 scanf 和 printf 函数功能。

需要说明的是，为了能突出 C++与 C 语言本身的不同，对于以往 C 语言的标准头文件（.h）也改用新的文件，去掉了“.h”扩展名。这就是说，【例 Ex_Simple】代码中头文件 iostream.h 的包含指令建议写成下面的新格式：

```
#include <iostream>
```

同时为使 iostream 中的定义对程序有效，还需使用下面名称空间编译指令来指定：

```
using namespace std;                    // 注意不要漏掉后面的分号
```

它是一个在代码编译之前处理的指令，namespace 称为**名称空间**。这样，C++程序基本框架建议为：

```
#include <iostream>
using namespace std;
int main()
{    …
     return 0;
}
```

事实上，cin 和 cout 就是 std 中已定义的流对象，若不使用“using namespace std;”，则还可有下列两种方式来指定。

1）在使用前用下列代码来指定：

```
using std::cout;                        // 指定以后的程序中可以使用 cout 对象
using std::cin;                         // 指定以后的程序中可以使用 cin 对象
```

2）在调用时指定它所属的名称空间，即如下述格式来使用：

```
std::cout<<"输入圆的半径: ";            // ::是域作用运算符，表示 cout 是 std 域中的成员
std::cin>>r;
```

1.2　程序书写规范

C++程序的书写规范基本与 C 相同。下面从标识符命名、缩进和注释这几个方面进行简单介绍。

1. 标识符命名

标识符是用来标识变量名、函数名、数组名、类名、对象名、类型名、文件名等的有效字符序列。标识符命名需要遵守合法性、有效性和易读性的原则。

（1）合法性

C++规定标识符由大小写字母、数字字符（0～9）和下划线组成，且第一个字符必须为字母或下划线。任何标识符中都不能有空格、标点符号、运算符及其他非法字符。标识符的大小写是有区别的，并且不能和系统的关键字同名，以下是常用的 C++标准关键字（C 语言本身有

32 个，斜体为 C++新增加的）：

asm	auto	*bool*	break	case	*catch*
char	*class*	const	continue	default	*delete*
do	double	else	enum	extern	float
for	*friend*	goto	if	*inline*	int
long	*mutable*	*namespace*	*new*	*operator*	*private*
protected	*public*	register	return	*short*	*signed*
sizeof	static	struct	switch	*template*	*this*
throw	*try*	typedef	union	unsigned	*using*
virtual	void	volatile	while		

（2）有效性

虽然，标识符的长度（组成标识符的字符个数）是任意的，但最好不要超过 32 个，因为有的编译系统只能识别前 32 个字符，也就是说前 32 个字符相同的两个不同标识符被有的系统认为是同一个标识符。

（3）易读性

在定义标识符时，若能做到“见名知意”就可以达到易读性的目的。

另外，为了增强程序的可读性，许多程序员采用“匈牙利标记法”来定义标识符。这种方法是：在每个变量名前面加上表示数据类型的小写字符，变量名中每个单词的首字母均大写。例如：用 nWidth 或 iWidth 表示整型（int）变量。

2. 缩进和注释

程序在书写时不要将程序的每一行都由第一列开始，而应在语句前面加进一些空格，称为“缩进”，或是在适当的地方加进一些空行，以提高程序的可读性。在本书中，采用下列缩进形式：

每个花括号占一行，并与使用花括号的语句对齐。花括号内的语句采用缩进书写格式，缩进量为四个字符（一个缺省的制表符）。

同缩进的目的一样，注释也是为了提高程序的可读性。注释本身对编译和运行并不起作用。在程序中，凡是放在“/*.....*/”之间或以“//”开头的行尾的内容都是注释的内容，其中，/*.....*/注释方式可以出现在程序中的任何位置。一般来说，注释应在编程的过程中进行，且注释内容一般有：源程序的总体注释（文件名、作用、创建时间、版本、作者及引用的手册、运行环境等）、函数注释（目的、算法、使用的参数和返回值的含义、对环境的一些假设等）及其他的少量注释。一般不要陈述那些一目了然的内容，以免影响注释的效果。

1.3　数据类型

和 C 语言一样，C++的数据类型也分为基本类型、派生类型以及复合类型三类。基本数据类型是 C++系统的内部数据类型，如整型、浮点型等。派生类型是将已有的数据类型定义成指针或引用。而复合类型是根据基本类型和派生类型定义的复杂数据类型（又可称为构造类型），如数组、类、结构体和共用体等。

1.3.1　基本数据类型

C/C++的基本数据类型有字符型（char）、整型（int）和浮点型（float、double）三种。这些基本数据类型还可用 short、long、signed 和 unsigned 来修饰。表 1-1 列出了 C++中的基本数据

类型，其字宽（以字节数为单位）和取值范围是在 Visual C++ 6.0 时的情况。

需要注意的是：

1）C++还可以有布尔型（bool），即值为 true 或 false。事实上，在计算机内，编译系统将 true 表示成整数 1，false 表示成整数 0，因此也可把布尔型看成是一个整型。

2）无符号（unsigned）和有符号（signed）的区别在于数值最高位的含义。对于 signed 类型来说，最高位是符号位，其余各位表示数值大小；而 unsigned 类型的各个位都用来表示数值大小；因此相同基本数据类型的 signed 和 unsigned 的数值范围是不同的。例如，无符号字符型值的范围为 0 ~ 255，而有符号字符型值的范围为-128 ~ -127。

3）char、short、int 和 long 可统称为整型。缺省时，char、short、int 和 long 本身是有符号（signed）的。

表 1-1　C++的基本数据类型

类　型　名	类 型 描 述	字　宽	范　围
char	字符型	1	-128 ~ 127
unsigned char	无符号字符型	1	0 ~ 255(0xff)
signed char	有符号字符型(与 char 相同)	1	-128 ~ 127
short [int]	短整型	2	-32 768 ~ 32 767
unsigned short [int]	无符号短整型	2	0 ~ 65 535(0xffff)
signed short [int]	有符号短整型(与 short int 相同)	2	-32 768 ~ 32 767
int	整型	4	-2 147 483 648 ~ 2 147 483 647
unsigned [int]	无符号整型	4	0 ~ 4 294 967 295(0xffffffff)
signed [int]	有符号整型(与 int 相同)	4	-2 147 483 648 ~ 2 147 483 647
long [int]	长整型	4	-2 147 483 648 ~ 2 147 483 647
unsigned long [int]	无符号长整型	4	0 ~ 4 294 967 295(0xffffffff)
signed long [int]	有符号长整型(与 long int 相同)	4	-2 147 483 648 ~ 2 147 483 647
float	单精度浮点型	4	3.4E-38 ~ 3.4E+38
double	双精度浮点型	8	1.7E-308 ~ 1.7E+308
long double	长双精度浮点型	8	1.7E-308 ~ 1.7E+308

注：表中[int]表示可以省略，即在 int 之前有 signed、unsigned、short、long 时，可以省略 int 关键字。

1.3.2　常　量

根据程序中数据的可变性，数据可以分为常量和变量两大类。

在程序运行过程中，其值不能被改变的量称为常量。常量可分为不同的类型，如 1、20、0、-6 为整型常量，1.2、-3.5 为浮点型常量，‘a’、‘b’为字符常量。常量一般从其字面形式即可判别。

下面简单介绍几种不同数据类型常量的表示方法。

1. 整型常量

整型常量可以用十进制、八进制和十六进制来表示。

十进制整型常量即十进制整数，如 34、128 等。

八进制整型常量是以 0 开头的数，它由 0 至 7 的数字组成。如 045，即$(45)_8$，表示八进制数 45，等于十进制数 37；-023 表示八进制数-23，等于十进制数-19。

十六进制整型常量是以 0x 或 0X 开头的数，它由 0 至 9、A 至 F 或 a 至 f 组成。例如 0X7B，即$(7B)_{16}$，等于十进制的 123，-0x1a 等于十进制的-26。

需要注意的是：

1）整型常量中的长整型（long）要以 L 或小写字母 l 作为结尾，如 3276878L、496l 等。

2）整型常量中的无符号型（unsigned）要以 U 或 u 作为结尾，如 2100U、6u 等。

3）整型常量中的无符号长整型（unsigned long）要以 U（或 u）和 L（或 l）的组合作为结尾，如 23UL、23ul、23LU、23lu、23Ul、23uL 等。

4）默认时，如果一个整数没有后缀，则可能是 int 或 long 类型，这取决该整数的大小。

2. 浮点型常量

浮点型常量即实数，它有十进制数或指数两种表示形式。

十进制数形式是由整数部分和小数部分组成的（注意必须有小数点）。例如 0.12、.12、1.2、12.0、12.、0.0 都是十进制数形式。

指数形式采用科学表示法，它能表示出很大或很小的浮点数。例如 1.2e9 或 1.2E9 都代表 1.2×10^9，注意字母 E（或 e）前必须有数字，且 E（或 e）后面的指数必须是整数。

若浮点型常量是以 F（或 f）结尾的，则表示单精度类型（float）；以 L（或小写字母 l）结尾的，表示长双精度类型（long double）。若一个浮点型常量没有任何说明，则表示双精度类型（double）。

3. 字符常量

字符常量是用单引号括起来的一个字符。如‘A’、‘g’、‘%’、‘␣’（␣表示空格，以下同）等都是字符常量。注意‘B’和‘b’是两个不同的字符常量。

除了上述形式的字符常量外，C/C++还可以用一个“\”开头的字符来表示特殊含义的字符常量。例如前例中‘\n’，代表一个换行符，而不是表示字母 n。这种将反斜杠(\)后面的字符转换成另外意义的方法称为转义表示法，‘\n’称为转义字符。表 1-2 列出了常用的转义字符。

表 1-2　常用转义字符

字 符 形 式	含　义
\a	响铃
\b	退格(相当于按 Backspace 键)
\f	进纸(仅对打印机有效)
\n	换行
\r	回车(相当于按 Enter 键)
\t	水平制表(相当于按 Tab 键)
\v	垂直制表(仅对打印机有效)
\'	单引号
\"	双引号
\\	反斜杠
\?	问号
\ooo	1 到 3 位八进制数所代表的字符
\xhh	1 到 2 位十六进制数所代表的字符

需要说明的是，当转义字符引导符后接数字时，用来指定字符的 ASCII 码值。默认时，数字为八进制，此时数字可以是 1 位、2 位或 3 位。若采用十六进制，则需在数字前面加上 **X** 或 **x**，此时数字可以是 1 位或多位。例如：‘\101’和‘\x41’都是表示字符‘A’。若为‘\0’，则表示 ASCII 码值为 0 的字符。

注意：ANSI/ISO C++中由于允许出现多字节编码的字符，因此对于“\x”或“\X”后接的十六进制的数字位数已不再限制。

4. 字符串常量

和 C 语言一样，C++除了允许使用字符常量外，还允许使用字符串常量。字符串常量是一对双引号括起来的字符序列。例如：

```
"Hello, World!\n"
"C++语言"
"abcdef"
```

等等都是合法的字符串常量。字符串常量的字符个数称为字符串长度。字符串常量中还可以包含空格、转义字符或其他字符。并且必须在同一行书写，若一行写不下，则需要用‘\’来连接，例如：

```
"ABCD\
EFGHIGK..."
```

不要将字符常量和字符串常量相混淆，它们的主要区别如下：

1）字符常量是用单引号括起来的，仅占一个字节；而字符串常量是用双引号括起来的，至少占用两个字节。例如‘a’是字符常量，占用一个字节用来存放字符 a 的 ASCII 码值，而“a”是字符串常量，它的长度不是 1 而是 2，除了字符 a 之外，它的末尾还有个‘\0’字符。每个字符串的末尾都有一个这样的字符，一定要注意。

2）字符常量实际上是整型常量的特殊形式，它可以参与常用的算术运算；而字符串常量则不能。例如：

```
int  b='a'+3;      // 结果 b 为 100，这是因为'a'的 ASCII 码值 97 参与了运算
```

5. 符号常量

在 C++中，除了用 C 语言的#define 定义符号常量外，还常常用 const 来定义符号常量。

1.3.3 变　量

变量是指在程序执行中其值可以改变的量。变量的作用是存储程序中需要处理的数据，它可以放在程序中的任何位置上。但无论如何，在使用一个变量前必须先定义这个变量。变量有三个基本要素：C++合法的变量名、变量类型和变量的数值。

1. 变量的定义

定义变量是用下面的格式语句进行定义的：

```
<类型> <变量名列表>;
```

需要说明的是：

1）可以将同类型的变量定义在一行语句中，不过变量名要用逗号(,)分隔。但在同一个程序块中，不能有两个相同的变量名。

2）注意在 C++中没有字符串变量类型，字符串是用字符类型的数组或指针来定义的。

3）与 C 语言相比，C++变量的定义比较自由。例如，可以在 for 语句中定义一个变量（以后还会讲到）。

2. 变量的初始化

程序中常需要对一些变量预先设置初值，这一过程称为初始化。在 C/C++中，可以在定义变量时同时使变量初始化。例如：

```
double   x=1.28;                    // 指定 x 为双精度浮点变量，初值为 1.28
int      nNum1, nNum2=3, nNum3;     // 指定某一个变量的初值
```

C++变量的初始化还有另外一种形式，它与 C 语言不同。例如：

```
int nX(1), nY(3);
```

表示 nX 和 nY 是整型变量，它们的初值分别为 1 和 3。

1.3.4　数据类型转换

C/C++采用两种方法对数据类型进行转换，一种是“自动转换”，另一种是“强制类型转换”。

1. 自动转换

自动转换是将数据类型从低到高的顺序进行转换，如图 1-1 所示。

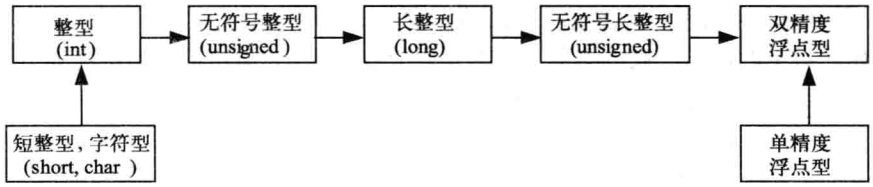


图 1-1　类型转换的顺序

2. 强制类型转换

强制类型转换是在程序中通过指定数据类型来改变图 1-1 所示的类型转换顺序，将一个变量从其定义的类型改变为另一种新的类型。强制类型转换有下列两种格式：

```
(<类型名>)<表达式>
<类型名>(<表达式>)
```

这里的“类型名”是任何合法的 C/C++数据类型，例如 float、int 等。通过类型的强制转换可以将“表达式”转换成指定的类型。

1.3.5　数组

C/C++可以允许程序员按一定的规则进行数据类型的构造，如定义数组类型、枚举类型、结构体类型和共用体类型等，这些类型统称为构造类型。

数组是相同类型的元素的有序集合，每个元素在数组中的位置可用统一的数组名和下标来唯一确定。

1. 数组的定义

定义一个数组可按下列格式进行：

```
<类型> <数组名> [<常量表达式 1>][<常量表达式 2>]…
```

<数组名>后面的常量表达式用于确定数组的维数和大小。例如：

```
int     a[10];
float   b[2][3];
```

分别定义了整型一维数组 a 和浮点型二维数组 b。

一般地，表示某维大小的常量表达式中不能包含变量，但可以包括常量和符号常量，其值必须是一个确定的整型数值，且数值大于 1。

2. 数组元素的引用

数组定义后，就可以引用数组中的元素，引用时按下列格式：

```
<数组名> [<下标>].....
```

例如 a[0]、b[5]等，这里的 0 和 5 是数组的下标，a 和 b 是定义过的数组名。

3. 数组的赋值

数组中的元素既可以在数组定义的同时赋初值，即初始化，也可以在定义后赋值。例如：

```
int  b[5]={1, 2};
```

是将数组 b 的元素 b[0]、b[1]分别赋予 1、2 的值。有时，在对全部数组元素赋初值时，可以不指定数组的长度，例如：

```
int  c[]={1, 2, 3, 4, 5};
```

系统将根据数值的个数自动定义 c 数组的长度，这里是 5。

对于二维或多维数组也可同样进行初始化。需要说明的是：

1）初始化数组的值的个数不能多于数组元素个数，初始化数组的值也不能通过跳过逗号的方式来省略，这在 C 中是允许的，但在 C++中是不允许的。例如：

```
int f[5] = {1, , 3, 4, 5};          // 错误，初始化值不能省略
int g[5] = {1, 2, 3, };             // 在 Visual C++中正确，它忽略最后一个值的逗号
int h[5] = { };                     // 语法格式错误
```

2）对于二维数组来说，如果对全部元素都赋初值，则定义数组时对第一维的大小可以忽略，但第二维的大小不能省。例如：

```
int k[][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
// 它等价于 int k[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

3）只对部分元素赋初值，可有两种说明方式：一种是以“行”为单位，依次列出部分元素的值；另一种是以数组元素的排列顺序依次列出前面部分元素的值。例如：

```
int m[3][4] = {{1, 2}, {3}, {4, 5, 6}};
```

没有明确列举元素值的元素，其值均为 0，即等同于：

```
int m[3][4] = {{1, 2, 0, 0}, {3, 0, 0, 0}, {4, 5, 6, 0}};
```

又如：

```
int n[3][4] = {1, 2, 3};
```

即 n[0][0] = 1，n[0][1] = 2，n[0][2] = 3，其余的各个元素的初值为 0。

【例 Ex_ArraySort】 把 5 个整数按从小到大的次序排列

```
#include <iostream.h>
const int ARRAYMAX=5;     // 用const定义一个符号常量
void main()
{
    int a[ARRAYMAX]={20, 40, -50, 7, 13};
    // 每次取余下数据的最小一个
    for(int i=0; i<ARRAYMAX; i++){
        for(int j=i+1; j<ARRAYMAX; j++){
            if(a[i]>a[j]){
                // 交换数据
                int temp = a[j];    a[j] = a[i];    a[i] = temp;
            }
        }
        cout<<a[i]<<" ";
    }
}
```

结果如下：

```
-50  7  13  20  40
```

4. 字符数组

在 C/C++语言中，一个字符串是用一个以空字符‘\0’作为结束符的字符串来表示的，例如：

```
char  ch[]={"Hello!"};                          // 第一种形式
```

或

```
char  ch[ ]="Hello!";                           // 第二种形式
```

或

```
char  ch[]={'H','e','l','l','o','!','\0'};      // 第三种形式
```

都是使得 ch[0]为‘H’，ch[1]为‘e’，ch[2]为‘l’，ch[3]为‘l’，ch[4]为‘o’，ch[5]为‘!’，ch[6]为‘\0’。但第二种形式是最简捷的。

上述定义的字符数组没有指定数组长度的目的是，避免在初始化时，字符串中的字符个数大于数组长度，从而产生语法错误；但如果指定的数组长度大于字符串中的字符个数，那么其余的元素将被系统默认为空字符‘\0’。例如：

```
char  ch[9] = "Hello!";
```

因“Hello!”的字符个数为 6，但还要包括一个空字符‘\0’，故数组长度至少是 7，从 ch[6]开始到 ch[8]都等于空字符。一定要小心字符数组与其他数组的这种区别，例如：

```
char  ch[6] = "Hello!";
```

虽然该代码不会引起编译错误，但由于改写了数组空间以外的内存单元，因此是危险的。正因为这一点，Visual C++将其列为数组超界错误。

对于二维的字符数组，其初始化也可依上进行。例如：

```
char  str[3][]={"How", "are", "you"};
```

这时，数组元素 str[0][0]表示一个字符，值为‘H’；但 str[0][ ]却表示一个字符串“How”，因为 str[0][ ]是一个一维字符数组。

1.3.6　结构体

一个结构体是由多种类型的数据组成的整体。组成结构的各个分量称为结构体的数据成员（简称为成员）。结构体是 C/C++构造复杂数据类型的手段之一。

1. 定义结构体

结构体定义的格式为：

```
struct <结构体名>
{
      <成员定义 1>;
      <成员定义 2>;
      ...
      <成员定义 n>;
} [结构体变量名列表];
```

结构体定义是以关键字 struct 作为标志的，<结构体名>应是一个有效的标识符。在结构体中的每个成员都必须通过“成员定义”来确定成员名及其类型。例如：

```
struct PERSON
{
      char    name[25];     // 姓名
      int     age;          // 年龄
      char    sex;          // 性别
```